```python
import numpy as np
import matplotlib.pyplot as plt
import useful as use
import pdb
from decimal import Decimal
np.set_printoptions(threshold=np.inf)


fig1, ax3 = plt.subplots()
fig2, ax4 = plt.subplots()
fig3, ax5 = plt.subplots()
###############################
#initial conditions


dt = .001

#############################################
#Theoretical Solution

l = [1]
r_0 = 1
v_0 = 0
t = 0
r = r_0
v = v_0
s = 2**(1/3)
w_0 = 1
gamma=.6
w = np.sqrt(w_0**2-gamma**2)
timing = [0]

for i in range(0,int(2*2*np.pi/dt)):

	r = np.e**(-gamma*t)*(r_0*np.cos(w*t)+(v_0+gamma*r_0)/w*np.sin(w*t))
	
	t+=dt
	l.append(r)
	timing.append(t+dt)

ax3.plot(timing,l)
ax4.plot(timing,l)

# err_theory = []
# count = 0
# for i in range(len(l)):

	# test = count/9000
	# if Decimal(test) % Decimal(0):
		# err_theory.append(l[i])
	# count+=1
# print(err_theory)

##############################################################
##############################################################
##############################################################
#First Order

dt = .9
l = [1]
r_0 = 1
v_0 = 0
t = 0
w_0 = 1
gamma=.6
w = np.sqrt(w_0**2-gamma**2)
timing = [0]
```

```
r = r_0
v = v_0
s = 2**(1/3)

for i in range(0,int((2*2*np.pi)/dt)):

    r = r+v*dt
    v = v - w_0**2*r*dt
    v = np.e**(-2*gamma*dt)*v
    l.append(r)
    timing.append(timing[len(timing)-1]+dt)

r = np.asarray(r)
ax3.plot(timing,l)

#################################################################
#################################################################
#Second Order
l = [1]
r_0 = 1
v_0 = 0
t = 0
w_0 = 1
gamma=.6
w = np.sqrt(w_0**2-gamma**2)
timing = [0]
r = r_0
v = v_0
s = 2**(1/3)

for i in range(0,int((4*np.pi)/dt)):

    v = v - w_0**2*r*.5*dt
    v = np.e**(-2*gamma*.5*dt)*v
    r = r+v*dt
    v = np.e**(-2*gamma*.5*dt)*v
    v = v - w_0**2*r*.5*dt

    l.append(r)
    timing.append(timing[len(timing)-1]+dt)

ax3.plot(timing,l)

#################################################################
#################################################################
#Fourth Order
l = [1]
r_0 = 1
v_0 = 0
t = 0
w_0 = 1
gamma=.6
w = np.sqrt(w_0**2-gamma**2)
r = r_0
v = v_0
s = 2**(1/3)
H = dt/(2-s)
timing = [0]

for i in range(0,int((4*np.pi)/dt)):

    v = v - w_0**2*r*.5*H
    v = np.e**(-2*gamma*.5*H)*v
    r = r+v*H
    v = np.e**(-2*gamma*.5*H)*v
    v = v - w_0**2*r*.5*H

```

```
	v = v - w_0**2*r*.5*-s*H
	v = np.e**(-2*gamma*.5*-s*H)*v
	r = r+v*-s*H
	v = np.e**(-2*gamma*.5*-s*H)*v
	v = v - w_0**2*r*.5*-s*H


	v = v - w_0**2*r*.5*H
	v = np.e**(-2*gamma*.5*H)*v
	r = r+v*H
	v = np.e**(-2*gamma*.5*H)*v
	v = v - w_0**2*r*.5*H


	l.append(r)
	timing.append(timing[len(timing)-1]+dt)

ax4.plot(timing,l)
ax4.set_ylabel('Particle Displacement')
ax4.set_xlabel('Time')
ax4.legend(('Theory','4th Order'), loc='upper right')
ax4.set_title("Damped HO - Comparison with .9 Timestep", va='bottom')


##################################################
##################################################
#graphing
ax3.set_ylabel('Particle Displacement')
ax3.set_xlabel('Time')
ax3.legend(('Theory', '1st Order', '2nd Order', '4th Order'), loc='upper right')
# ax3.plot([-.5,.5], [0,0], 'o', markerfacecolor='none', markeredgecolor='r')
ax3.set_title("Damped HO - Comparison with .9 Timestep", va='bottom')
plt.show()
```